Indexing und Performance Tuning

Cybertec Schönig & Schönig GmbH
Hans-Jürgen Schönig

PostgreSQL Indexing

- Jeder hat schon einmal ein Telefonbuch Benutzt

- Jeder hat schon einmal Suchen durchgeführt

CREATE INDEX

test=# \h **CREATE INDEX**
Command: CREATE INDEX
Description: define a new index
Syntax:
CREATE [UNIQUE] INDEX [CONCURRENTLY] [name] ON table [USING method]
({ column | (expression) } [COLLATE collation] [opclass] [ASC | DESC] [NULLS { FIRST | LAST }] [, ...])
[WITH (storage_parameter = value [, ...])]
[TABLESPACE tablespace]
[WHERE predicate]

Mehr als nur ein Baum

- PostgreSQL unterstützt 4 Arten von Indexes:

  => B-Tree

  => Gist
  => GIN
  => Hashes

Wieso verschiedene?

- Nicht jeder “Algorithmus” passt für jede Art der Abfrage

- Verschiedene Indexes erfüllen verschiedene Aufgaben

CYBERTEC
The PostgreSQL Database Company

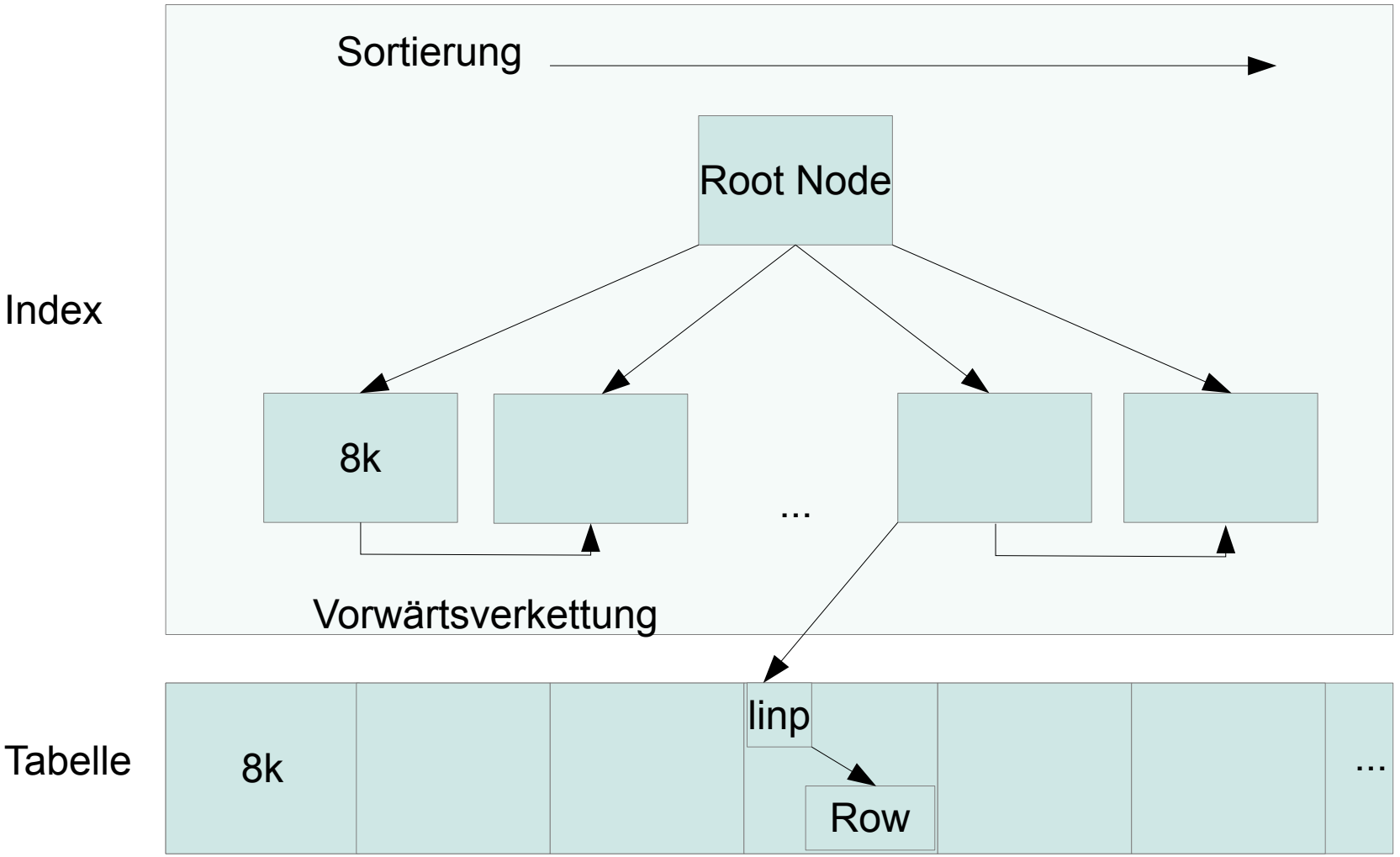
B-Tree (1)

- B-Bäume sind die “typischen” Indices

- Wie ein “Telefonbuch”

  => sortiert
  => in “Pages” gespeichert

B-Tree (2)


Sortierung
Root Node
Index
8k
...
Vorwärtsverkettung
Tabelle
8k
linp
Row
...

B-Tree (3)

```
test=# CREATE TABLE t_test AS SELECT *
     FROM generate_series(1, 100000) AS x;
SELECT 100000

test=# CREATE INDEX idx_x ON t_test (x);
CREATE INDEX

test=# EXPLAIN SELECT * FROM t_test ORDER BY x LIMIT 10;
                              QUERY PLAN
-----------------------------------------------------------------------
 Limit  (cost=0.00..0.44 rows=10 width=4)
   ->  Index Scan using idx_x on
         t_test  (cost=0.00..4376.26 rows=100000 width=4)
(2 rows)
```

CYBERTEC
The PostgreSQL Database Company

Gist (1)

- Wie sortiert man Gartenteiche?

CYBERTEC
The PostgreSQL Database Company

Gist (2)

- Geodaten sind nicht sinnvoll sortierbar

  => B-Tree ist folglich ungeeignet

- alternativ dazu arbeitet man eher mit “contains” als mit “<” und “>”.

CYBERTEC
The PostgreSQL Database Company

Gist (3)

- Gist verwendet man für ...

  => Geodaten

  => Volltextsuche

  => etc.

KNN-Suche

- spezielle Traversierungsstrategie ermöglicht “Umkreissuche” via Gist

- Beispiel:

```
test=# SELECT x, 1000 <-> x AS dist
          FROM t_test ORDER BY 1000 <-> x LIMIT 5;
   x   | dist
  -----+------
  1000 |    0
  1001 |    1
   999 |    1
  1002 |    2
   998 |    2
(5 rows)
```

GIN / Hashes

- Hashes sind in der Praxis eher selten

- GIN (“Inverted Index”) kann für Volltextsuche verwendet werden.

Wenn das nicht reicht?

- PostgreSQL erlaubt die Definition sogenannter “Operator Classes”

- Definition eigener Indizierungsstrategien

CYBERTEC
The PostgreSQL Database Company

Operator classes (1)

- Beispiele:

  => Wie sucht man in Genomen?

  => Wie findet man ähnliche Farben?

  => Wir sortiert man Produktnummern?

  => Sollen Eilzüge vor Regionalzügen gereiht sein?

CYBERTEC
The PostgreSQL Database Company

Operator classes (2)

- Jeder Index hat unterschiedliche “Strategien”, die man je nach Bedarf bauen kann.

CYBERTEC
The PostgreSQL Database Company

Operator classes (3)

- B-Tree:

less than	1
less than or equal	2
equal	3
greater than or equal	4
greater than	5

CYBERTEC
The PostgreSQL Database Company

Operator classes (4)

- Gist:

strictly left of	1
does not extend to right of	2
overlaps	3
does not extend to left of	4
strictly right of	5
same	6
contains	7
contained by	8
does not extend above	9
strictly below	10
strictly above	11
does not extend below	12

Viele Indices

- Mehr als ein Index pro Abfrage:

```
test=# explain SELECT * FROM t_test WHERE x = 3 OR x = 4;
                          QUERY PLAN
-----------------------------------------------------------------------
 Bitmap Heap Scan on t_test  (cost=8.54..16.16 rows=2 width=4)
   Recheck Cond: ((x = 3) OR (x = 4))
   ->  BitmapOr  (cost=8.54..8.54 rows=2 width=0)
         ->  Bitmap Index Scan on idx_x  (cost=0.00..4.27 rows=1 width=0)
               Index Cond: (x = 3)
         ->  Bitmap Index Scan on idx_x  (cost=0.00..4.27 rows=1 width=0)
               Index Cond: (x = 4)
(7 rows)
```

Functional Indexing

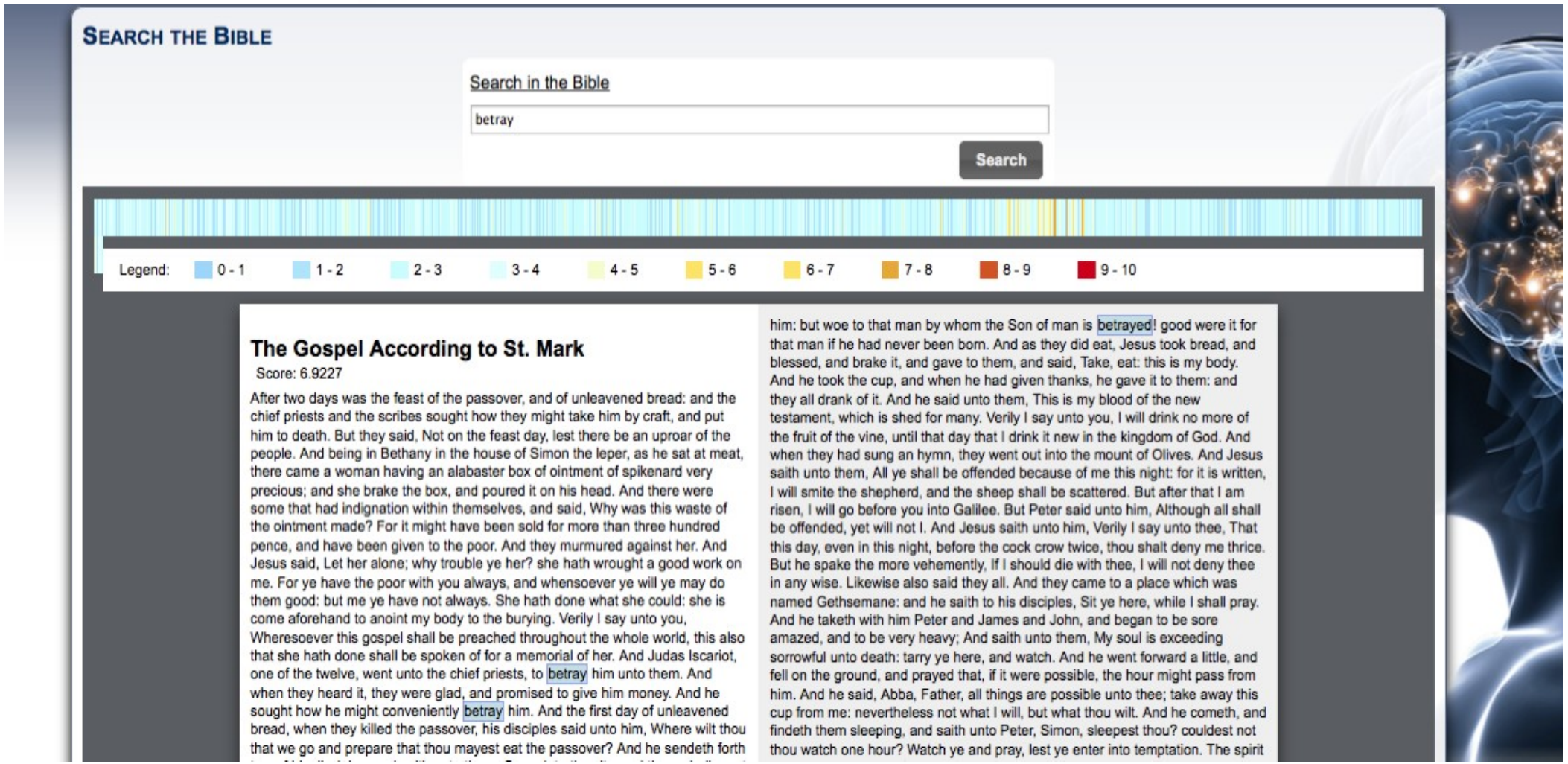
- Indexes auf Funktionen

```
test=# CREATE INDEX idx_cos ON t_test ( cos(x) );
CREATE INDEX

test=# explain SELECT * FROM t_test WHERE cos(x) = 0;
                          QUERY PLAN
---------------------------------------------------------------------
 Index Scan using idx_cos on t_test  (cost=0.00..8.28 rows=1 width=4)
   Index Cond: (cos((x)::double precision) = 0::double precision)
(2 rows)
```

CYBERTEC
The PostgreSQL Database Company

Putting it all together



Fazit

Herzlichen Dank für Ihre Aufmerksamkeit

www.postgresql-support.de
hs@cybertec.at

CYBERTEC
The PostgreSQL Database Company

Finally ...

Wir suchen Mitarbeiter !

Bewerbungen bitte an
hs@cybertec.at